Journal of Science and Technology in Civil Engineering, HUCE (NUCE), 2021, 15 (4): 99–110

OPTIMIZATION OF STEEL ROOF TRUSSES USING MACHINE LEARNING-ASSISTED DIFFERENTIAL EVOLUTION

Nguyen Tran Hieu[a,*], Nguyen Quoc Cuong[a], Vu Anh Tuan[a]

[a] *Faculty of Buildings and Industrial Constructions, Hanoi University of Civil Engineering, 55 Giai Phong road, Hai Ba Trung district, Hanoi, Vietnam*

Article history:
Received 02/08/2021, Revised 16/09/2021, Accepted 17/09/2021

Abstract

A steel truss is a preferred solution in large-span roof structures due to its good attributes such as lightweight, durability. However, designing steel trusses is a challenging task for engineers due to a large number of design variables. Recently, optimization-based design approaches have demonstrated the great potential to effectively support structural engineers in finding the optimal designs of truss structures. This paper aims to use the AdaBoost-DE algorithm for optimizing steel roof trusses. The AdaBoost-DE employed in this study is a hybrid algorithm in which the AdaBoost classification technique is used to enhance the performance of the Differential Evolution algorithm by skipping unnecessary fitness evaluations during the optimization process. An example of a duo-pitch steel roof truss with a span of 24 meters is carried out. The result shows that the AdaBoost-DE achieves the same optimal design as the original DE algorithm, but reduces the computational cost by approximately 36%.

Keywords: structural optimization; Differential Evolution; machine learning; AdaBoost; steel roof truss.

https://doi.org/10.31814/stce.huce(nuce)2021-15(4)-09

1. Introduction

Steel trusses have been widely used in large-span buildings and constructions such as factory buildings, warehouses, hangars, gymnasiums, etc. In truss structures, the material is arranged far away from the neutral axis. Due to the reasonable distribution of material, truss structures often require fewer materials to span the same distance when compare with other options like beams or girders. Moreover, with the same amount of material, a truss can span longer than a beam, which means trusses are stronger than beams. Therefore, truss structures can be an affordable option in many cases. Although truss structures have many advantages as listed above, this kind of structure has also disadvantages, for example, it requires a lot of space as well as high maintenance costs. In addition, because truss structures are intricate, complex with many elements, a good design requires a lot of human resources. The traditional way to design steel structures is using the process of trial and error to search for a feasible solution that satisfies all design requirements. For large-scale structures with a huge number of members and a wide list of available profiles, the number of candidates that must be

[*]Corresponding author. *E-mail address:* hieunt2@nuce.edu.vn (Hieu, N. T.)

tried is too excessive and this method becomes inefficient. Recently, optimization-based design approaches have demonstrated the great potential to effectively support structural engineers in finding the optimal solution.

The most intuitive optimization method is the fully stressed design (FSD) in which the cross-sections of all members are iteratively updated until the stress in each member reaches the allowable stress value. However, the results found in [1] have indicated that at the optimal state, a member is not stressed to its limit. This conclusion violates the assumptions of the FSD method. Subsequent studies mostly used gradient-based optimization methods to solve structural optimization problems. Since the early work of Goldberg and Samtani in which the Genetic Algorithm was employed to optimize the weight of 10-bar truss structure [2], many meta-heuristic algorithms have been proposed and successfully applied to solve structural optimization problems. Numerous outstanding algorithms can be mentioned as: Genetic Algorithm (GA) [2], Evolution strategies (ES) [3], Differential Evolution (DE) [4], Particle Swarm Optimization (PSO) [5], Ant Colony Optimization (ACO) [6], Artificial Bee Colony (ABC) [7], Harmony Search (HS) [8], etc. Several studies related to the application of meta-heuristic algorithms to optimize steel roof trusses can be listed as follows: Koumousis and Georgiou using GA [9], Croce et al using GA [10], Hamza et al. using taboo search [11].

In general, by searching over a large space, meta-heuristic algorithms can often found the globally optimal solution but a lot of fitness evaluations are usually needed. Although the time for each fitness evaluation is not too long, the total computation time is very large because thousands of fitness evaluations must be performed. For example, each function evaluation for a steel truss frame takes only 4 seconds but the optimization time is up to 6.73 hours with 6060 times of function evaluation [12]. One effective solution to deal with this problem is to employ machine learning (ML) models to estimate the fitness of solutions generated during the optimization process [13–17]. In all the above-mentioned studies, ML models are built based on the regression neural network. It should be noted that besides Neural Network (NN), there are many other ML regression techniques such as Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), etc. These techniques have been well employed to predict the load-capacity of structures [18–21].

Different from [13–17], a novel technique is introduced in [22] when a classification neural network is employed with the aim of evaluating the safety state of structures. This model is embedded into the DE algorithm to reduce unnecessary fitness evaluations, thereby shorten the optimization time. However, some comparative studies shows that AdaBoost algorithm has the best performance among common ML algorithms in both predicting the behavior [23] and evaluating the safety state of trusses [24]. Therefore, in this paper, a hybrid algorithm called AdaBoost-DE that combines the AdaBoost classification technique and the DE algorithm is used to optimize the weight of a planar roof truss. A comparison is made between the numbers of fitness evaluations performed by the AdaBoost-DE and the original DE to exhibit the benefits of the proposed algorithm.

The paper is organized as follows. The formulation of the weight optimization of truss structures is presented in Section 2. Section 3 gives a brief introduction to the original DE and the AdaBoost-DE algorithms. An example of a roof truss with a span of 24 meters is carried out in Section 4. Finally, conclusions are given in Section 5.

2. Optimization of steel roof trusses

2.1. Design of steel roof trusses according to Vietnamese specifications

A truss is basically a triangulated system of members that are interconnected at nodes. The connections at nodes are often assumed to be nominally pinned, therefore, the principal force in each member is axial force. Trusses are categorized into two types: planar trusses and space trusses. A planar truss is a truss in which all members are in the same plane while a space truss is a three-dimensional system of connected triangles. In a typical factory building, a roof structure consists of a series of planar trusses linked together by lateral bracing (Fig. 1). Planar trusses are used to carry the roof loads while the bracing provides the longitudinal stability of the structure. It needs to provide either a single or a double slope to the upper chord for drainage purposes.

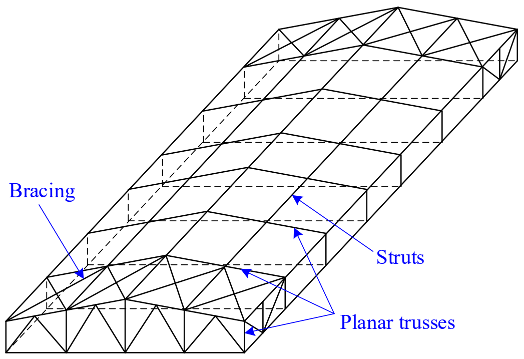

Figure 1. Arrangement of a typical roof structure

The design procedure for a steel truss according to the Vietnamese specifications contains 4 steps: (i) determination of the applied loads; (ii) structural analysis; (iii) checking of limit states including the ultimate limit state (ULS) and the serviceability limit state (SLS); and (iv) design of connections.

The loads acting on roof structures are taken according to TCVN 2737:1995 [25]. In more detail, there are four load types that must be considered when designing roof structures: the dead load (including self-weight of structures, the weight of roofing and equipment, dust, etc), the maintenance load, the wind load, and the seismic load. The structural analysis is then performed to determine the member forces and the nodal displacements. After that, the truss structure is checked based on two limit states. In the SLS, the maximum deflection of the truss structure is verified:

$$u \leq [u] \tag{1}$$

in which: u is the vertical displacement of the truss structure, $[u]$ is the allowable displacement for truss structures. According to TCVN 5575:2012 [26], the allowable displacement for truss structures $[u]$ equals $L/250$, where L is the span of the structure.

$$\frac{N}{A_n} \leq f\gamma_c \tag{2}$$

where: N is the design value of the tension force; A_n is the net cross-sectional area; f is the design strength of the steel material; γ_c is the working condition factor.

Compression members must be verified the buckling condition as follows:

$$\frac{N}{\varphi A} \leq f\gamma_c \tag{3}$$

in which: N is the design value of the compression force; φ is the buckling coefficient; A is the gross cross-sectional area; f is the design strength of the steel material; γ_c is the working condition factor.

The buckling coefficient φ can be determined using three equations as follows:

$$\text{If } 0 < \bar{\lambda} \leq 2.5: \qquad \varphi = 1 - \left(0.073 - 5.53\frac{f}{E}\right)\bar{\lambda}^{1.5} \tag{4a}$$

$$\text{If } 2.5 < \bar{\lambda} \leq 4.5: \qquad \varphi = 1.47 - 13\frac{f}{E} - \left(0.371 - 27.3\frac{f}{E}\right)\bar{\lambda} + \left(0.0275 - 5.53\frac{f}{E}\right)\bar{\lambda}^2 \tag{4b}$$

$$\text{If } \bar{\lambda} > 4.5: \qquad \varphi = \frac{332}{\bar{\lambda}^2(51 - \bar{\lambda})} \tag{4c}$$

in which: $\bar{\lambda} = \lambda\sqrt{f/E}$ is the conventional slenderness ratio; f and E is the design strength and the modulus of elasticity of the steel material, respectively; $\lambda = \mu l/r$ is the slenderness ratio of the member; μl is the buckling length of the member (Table 1); r is the radius of gyration of the cross-section of the member.

In addition, the maximum slenderness ratio is limited to $180 - 60\,(N/\varphi A f \gamma_c)$ and 400 for compression members and tension members, respectively.

Table 1. Effective lengths of truss members

Member	In-plane effective length l_x	Out-of-plane effective length l_y
Chords	Distance between two adjacent nodes	Distance between two longitudinal struts
Support web members	l	l
Web members	$0.8l$	l

2.2. *Formulation of the optimization problem*

The sizing optimization problem of a truss structure can be stated as follows:

$$\begin{aligned} &\text{find} && \mathbf{A} = \{A_i, i = 1, 2, \ldots, D\} \\ &\text{to minimize} && W(\mathbf{A}) = \sum_{i=1}^{n_m} \rho A_i L_i \\ &\text{subject to} && g_j(\mathbf{A}) \leq 0, j = 1, 2, \ldots, m \\ &&& A_i \in \mathbf{S} = \{S_1, S_2, \ldots, S_d\} \end{aligned} \tag{5}$$

where: $\mathbf{A}$ is a D-dimensional vector containing D design variables A_i; $W(\mathbf{A})$ is the objective function which is the weight of the truss in this case (kg); n_m is the number of truss members; ρ is the density (7850 kg/m^3 for the steel material); A_i is the cross-sectional area of the i-th member; L_i is the length of the i-th member; $g_j(\mathbf{A})$ is the j-th constraint; m is the number of constraints; $\mathbf{S}$ is the list containing d available profiles.

Constraints of the optimization problem are design requirements that are presented in Section 2.1 including ULS conditions, SLS conditions, and maximum slenderness limitations. These constraints can be rewritten in the normalized form as follows:

$$\begin{aligned} g_u(\mathbf{A}) &= \frac{u}{[u]} - 1 \leq 0 \\ g_\sigma(\mathbf{A}) &= \begin{cases} \dfrac{N_t}{A_n f \gamma_c} - 1 \leq 0 \\ \dfrac{N_c}{\varphi A f \gamma_c} - 1 \leq 0 \end{cases} \\ g_\lambda(\mathbf{A}) &= \frac{\lambda}{[\lambda]} - 1 \leq 0 \end{aligned} \tag{6}$$

To handle constraints, a widely used method called the penalty function is applied in this study. The fitness function is employed instead of the objective function by adding a very large term to penalize when there is any constraint violation. The mathematical formula of the fitness function is as follows:

$$F(\mathbf{A}) = W(\mathbf{A}) + PF \times \left\{1 + \max\left[\max_j\left(1, g_j(\mathbf{A})\right)\right]\right\} \tag{7}$$

in which: $F(\mathbf{A})$ is the fitness function; PF is a very large value number. Other parameters such as the applied loading, the configuration as well as the geometry of the truss are fixed.

3. The AdaBoost-DE algorithm

3.1. Differential Evolution algorithm

In this study, the DE algorithm is used to solve the weight optimization problem described in Section 2.2. The algorithm proposed by Storn and Price in 1995 contains four steps [27]: initialization, mutation, crossover, and selection. Many variants of the DE algorithm have been proposed. In this study, a powerful variant, called target-to-best/1 is employed. The procedure of the DE algorithm is presented as follows:

- Initialization: an initial population of NP vectors $\{\mathbf{x}_i^{(0)}, i = 1, 2, \ldots, NP\}$ is generated using the following operator:

$$x_{ij}^{(0)} = x_j^L + \text{rand}(0,1) \times \left(x_j^U - x_j^L\right) \tag{8}$$

where $x_{ij}^{(0)}$ is the$j$-th component of the D-dimensional vector $\mathbf{x}_i^{(0)}$; rand(0, 1) is a function that returns a uniformly distributed random real value in the range (0, 1); x_j^L and x_j^U are the lower and upper bounds of x_{ij}, respectively

- Mutation: at the t-th generation, a mutant vector $\mathbf{v}_i$ is created based on the target vector $\mathbf{x}_i$:

$$\mathbf{v}_i^{(t)} = \mathbf{x}_i^{(t)} + F \times \left(\mathbf{x}_{best}^{(t)} - \mathbf{x}_i^{(t)}\right) + F \times \left(\mathbf{x}_{r1}^{(t)} - \mathbf{x}_{r2}^{(t)}\right) \tag{9}$$

where: $r_1 \neq r_2 \neq i$ are randomly selected from the range $\{1, 2, \ldots, NP\}$; F is the scaling factor; $\mathbf{x}_{best}^{(t)}$ is the best vector of the t-th population.

- Crossover: a trial vector $\mathbf{u}_i$ is created from some components from the mutant vector $\mathbf{v}_i$ and the remaining ones from the target vector $\mathbf{x}_i$:

$$u_{ij}^{(t)} = \begin{cases} v_{ij}^{(t)} & \text{if } i = K \text{ or } \text{rand}[0,1] \le Cr \\ x_{ij}^{(t)} & \text{otherwise} \end{cases} \tag{10}$$

where: $\mathbf{u}_{ij}(t), \mathbf{v}_{ij}(t)$, and $\mathbf{x}_{ij}(t)$ are the j-th component of $\mathbf{u}_i(t), \mathbf{v}_i(t), \mathbf{x}_i(t)$, respectively; K is randomly chosen integer in the interval $[1, D]$; Cr is the crossover rate.

- Selection: the better one between the trial vector $\mathbf{u}_i(t)$ and the target vector $\mathbf{x}_i(t)$ will enter the next generation:

$$\mathbf{u}_i^{(t+1)} = \begin{cases} \mathbf{u}_i^{(t)} & \text{if } F\left(\mathbf{u}_i^{(t)}\right) \le F\left(\mathbf{x}_i^{(t)}\right) \\ \mathbf{x}_i^{(t)} & \text{otherwise} \end{cases} \tag{11}$$

in which: $F(.)$ is the fitness function that is defined in Section 2.2.

Three last operators are repeated (*max_iter*-1) times for obtaining the final results. Besides, the DE algorithm is originally designed for continuous variables. The problem stated in Section 2.2 is categorized as discrete optimization when the design variables are selected from a list. Therefore, a technique should be applied to handle discrete variables where the design variable A_i is converted to the sequence number I_i representing the position of A_i in the list **S**. Additionally, a rounding technique is employed to adjust a continuous value to a discrete value as follows:

$$x_{discrete} = \text{round}\ (x_{continuous}) \tag{12}$$

where: round(.) is a function that returns the nearest integer of a number.

3.2. Enhancement of the DE algorithm

In the selection step, the DE algorithm adopts a pairwise comparison in which each trial vector is only compared to the corresponding target vector. Moreover, it is clearly seen that the trial vector having a larger objective function value and violating any constraint is mostly worse than its target vector. Based on this observation, a modified DE algorithm is proposed in Ref. [22]. The neural network classifier is incorporated into the DE algorithm with the aim of evaluating whether the trial vector violates any constraint or not. The prediction of the neural network classifier aids in deciding whether to evaluate the trial vector with the true fitness function or not. It means not all trial vectors are evaluated with the fitness functions. As a result, some unnecessary fitness evaluations are reduced in comparison with the original DE algorithm. Nonetheless, a comparative study conducted in Ref. [24] indicates that the AdaBoost model seen to be more competitive than other classification models like NN, SVM in classifying the safety state of truss structures.

In the present work, a different algorithm called AdaBoost-DE is used to solve the structural optimization problem in which the AdaBoost classifier is employed instead of the neural network classifier. The AdaBoost is an ensemble method that was proposed by Freund and Schapire [28]. The characteristic of the AdaBoost is to create a strong classifier from several weak classifiers that are iteratively trained on the weighted dataset. The final prediction of the strong classifier is made by taking the weighted majority votes of the predictions of all weak classifiers. To avoid wordiness, the details of the AdaBoost are not presented in this paper. Readers can refer to Ref. [28]. A brief illustration of the AdaBoost is presented in Fig. 2.

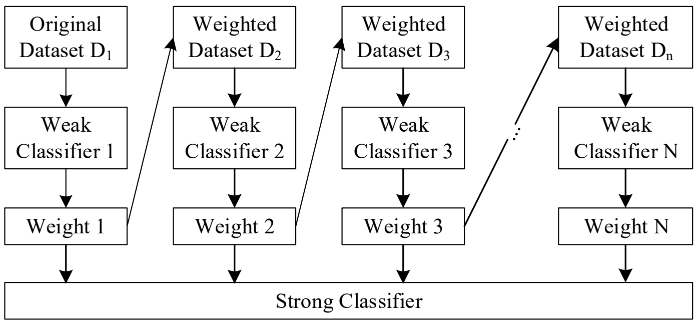

Figure 2. Illustration of the AdaBoost

In general, the AdaBoost-DE compromises two stages. The first stage performs the original DE algorithm with the aim of exploring new space. All trial vectors in the first stage are evaluated by the true fitness function. Each trial vector will be assigned a label "+1" if it satisfies all constraints, otherwise, the assigned label is "−1". Besides, a training dataset is collected from the historically evaluated vectors, and then an AdaBoost classifier is built on the obtained dataset. In the second stage, each trial vector will be preliminarily assessed using the AdaBoost classifier. At this time, one of three situations can occur as presented in Table 2.

Table 2. Three possible situations in the AdaBoost-DE

Situation	Predicted label	Comparison of the objective functions	Action
(i)	label = "+1"		Fitness evaluation
(ii)	label = "−1"	$W(\mathbf{u}) \leq W(\mathbf{x})$	Fitness evaluation
(iii)	label = "−1"	$W(\mathbf{u}) > W(\mathbf{x})$	Elimination

It is apparent from the table that some worse trial vectors are skipped without conducting true fitness evaluations. Consequently, the computation time can be shortened. The flowchart of the AdaBoost-DE algorithm is presented in Fig. 3.

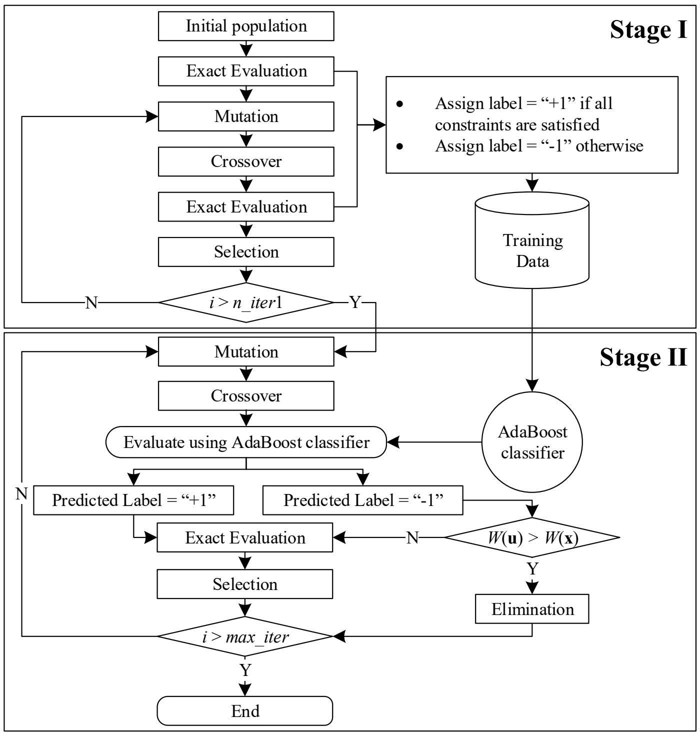

Figure 3. Flowchart of the AdaBoost-DE

4. Numerical example

4.1. Design data

A steel roof structure built in an industrial factory project in Bac Giang province is employed to demonstrate the practical applicability of the AdaBoost-DE algorithm. This structure is designed according to the Vietnamese specifications, and the material used in this project has the modulus of elasticity E = 200,000 N/mm^2 and the design strength f = 210 N/mm^2. The roof structure consists of several planar trusses with a span $L = 24.0$ m arranged parallel to a spacing $B = 7.0$ m as shown in Fig. 4. The height of the truss at the supports is $h_1 = 1.0$ m. For roof slope $i = 10\%$, the height at mid-span $h_2 = 2.2$ m. The longitudinal bracing is provided for ensuring lateral stability. The connections between trusses and R.C columns are designed as pinned supports.

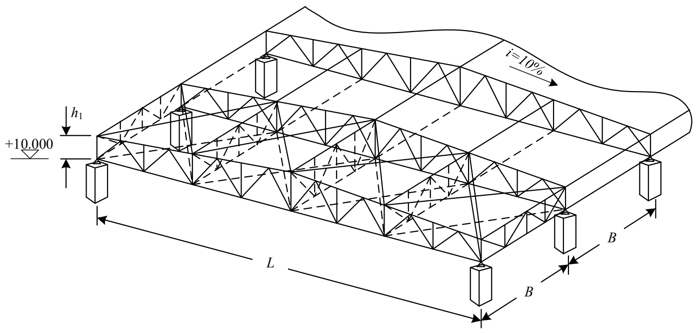

Figure 4. Industrial factory roof structure

The planar trusses used in this project are the Warren-type truss consisting of square hollow section (SHS) members with welded joints. Due to the limitation of Vietnamese profile types, this project uses profiles according to European standards. Each planar truss has 49 members that are categorized into five groups including: (1) top chords; (2) bottom chords; (3) support diagonals; (4) diagonals; (5) posts. Members of the same group are assigned the same cross-section which is selected from a list of 55 SHS profiles (from □50 × 3 to □300 × 12.5).

The roof structure is subjected to three load cases: the dead load (DL), the live load (LL), and the wind load (WL). The dead load includes the self-weight of the structure, the weight of purlins and sheeting ($g_1 = 15$ daN/m^2), and the weight of equipment ($g_2 = 50$ daN/m^2). The live load is taken as the maintenance load for unused sloping roofs $p = 30$ daN/m^2. The project is built in region II.B according to TCVN 2737:1995 with the basic wind pressure $W_0 = 95$ daN/m^2. The loads distributed on the roof are converted to the concentrated loads at the nodes of the truss as shown in Fig. 5. Three load combinations are considered: (LC1) DL+LL; (LC2) DL+WL; (LC3) DL+0.9×LL+0.9×WL.

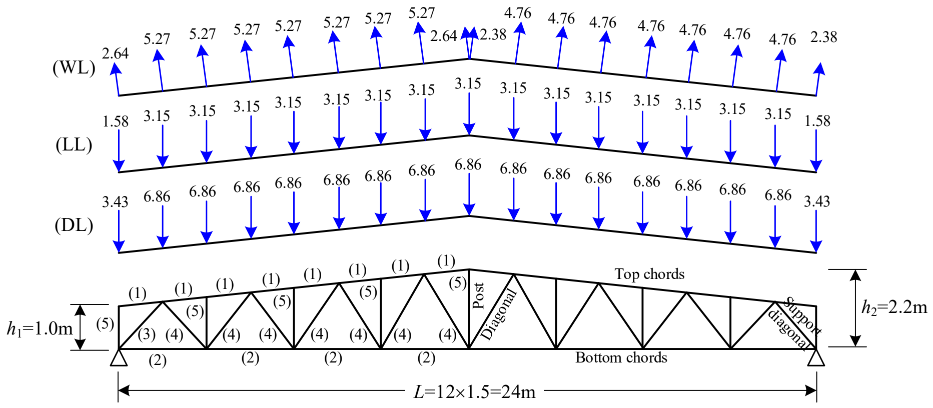

Figure 5. Configuration of the planar truss

4.2. Setting

The truss structure is optimized according to both the original DE algorithm and the AdaBoost-DE algorithm. For a fair comparison, the parameters of the two algorithms are set to the same as follows: the scaling factor $F = 0.8$, the crossover rate $Cr = 0.9$, the population size $NP = 50$, the number of generations $max\_iter = 100$. For the AdaBoost-DE, the first stage lasts $n\_iter1 = 10$ which means that the training dataset contains 500 samples. Two algorithms are coded in Python language with some widely used libraries such as scikit-learn. The finite element code for fitness evaluations is also written in Python based on the direct stiffness method. Each algorithm is performed 30 times on a computer with the following configuration: CPU Intel Core i5-5257 2.7 GHz, RAM 8.00 Gb.

4.3. Results

The statistical results of 30 optimization times are presented in Table 3. Additionally, the convergence histories of two algorithms are displayed in Fig. 6.

Table 3. Comparison of optimal results found by the DE and the AdaBoost-DE

	DE	AdaBoost-DE
(1) Top chords	□150×4	□150×4
(2) Bottom chords	□120×4	□120×4
(3) Support diagonals	□80×3	□80×3
(4) Diagonals	□70×2.5	□70×2.5
(5) Post	□50×3	□50×3
Best weight (kg)	1022.9	1022.9
Mean weight (kg)	1060.7	1065.1
Worst weight (kg)	1213.9	1213.9
Standard deviation (kg)	48.5	49.6
Average number of fitness evaluations	5000	3198

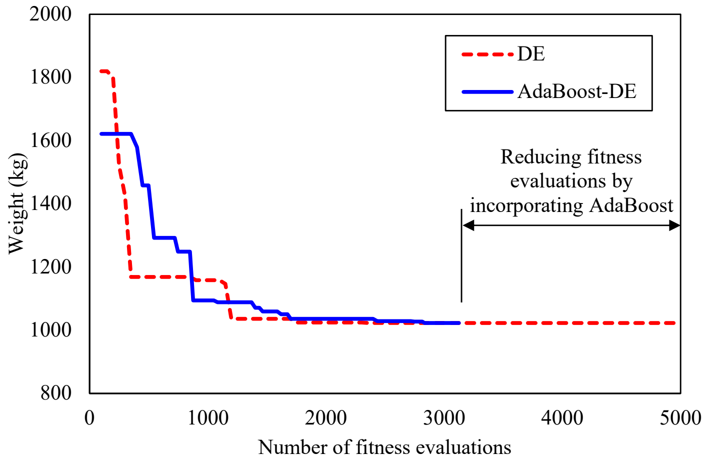

Figure 6. Convergence curves for the optimization of the roof truss

Based on the obtained results, some observations can be pointed out as follows. First of all, two algorithms DE and AdaBoost-DE achieve the same best weight (1022.9 kg) but the standard deviation (SD) of the DE is smaller than that of the AdaBoost-DE. It indicates that the DE is more stable than the AdaBoost-DE. However, the difference is not much (the SD of the DE is 48.5 kg while the SD of the AdaBoost-DE is 49.6 kg). Secondly, the AdaBoost-DE carries out fewer fitness evaluations than the DE (3198 times for the AdaBoost-DE and 5000 times for the DE). It means using AdaBoost helps to reduce approximately 36% fitness evaluations of the DE optimization.

4.4. Comparison of different truss types

The proposed method AdaBoost-DE is applied to optimize the weight for 06 cases with the same design data as described in Section 4.1 but the configurations of trusses are different. More specifically, six truss types considered in this section are as follows: Case 1: a trapeizoidal Warren truss (Fig. 7(a)), Case 2: a parallel chord Warren truss (Fig. 7(b)), Case 3: a trapeizoidal Pratt truss (Fig. 7(c)), Case 4: a parallel chord Pratt truss (Fig. 7(d)), Case 5: a trapeizoidal Howe truss (Fig. 7(e)), and Case 6: a parallel chord Howe truss (Fig. 7(g)). Parameters are set the same as Section 4.2. Each case is implemented for 15 independent runs. The best designs of six cases are reported in Table 4. Additionally, demand-to-capacity ratios for six truss cases are also summarized in this table.

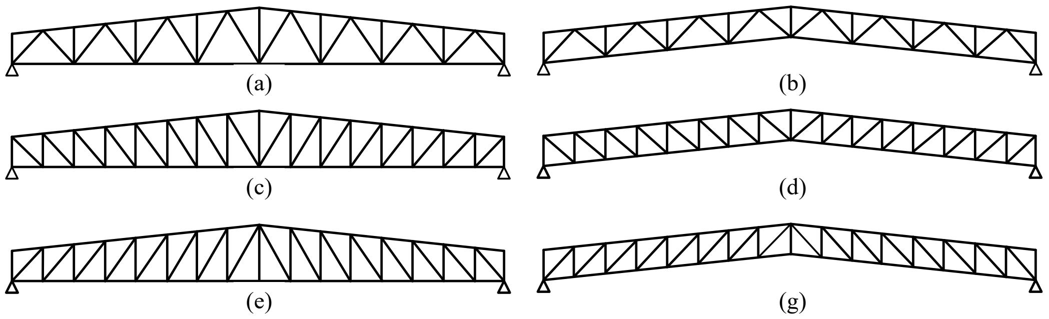

Figure 7. Configurations of six truss types

Table 4. Comparison of best designs for six truss types

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
(1) Top chords	□150×4	□150×4	□150×4	□150×4	□150×4	□150×4
(2) Bottom chords	□120×4	□150×4	□120×5	□150×5	□120×4	□150×4
(3) Support diagonals	□80×3	□70×2.5	□70×2.5	□50×3	□80×3	□70×2.5
(4) Diagonals	□70×2.5	□50×3	□50×3	□50×3	□70×2.5	□70×2.5
(5) Post	□50×3	□50×3	□50×3	□50×3	□50×3	□50×3
Weight (kg)	1022.9	1031.7	1118.2	1160.7	1081.8	1095.8
max $(N/Af\gamma_c, \lvert N/\varphi A f\gamma_c\rvert)$	0.933	0.915	0.977	0.890	0.937	0.928
max$(u/[u])$	0.303	0.289	0.374	0.317	0.294	0.268
max$(\lambda/[\lambda])$	0.849	0.677	0.892	0.684	0.849	0.677

It is clearly seen that Case 1 has the lowest weight (1022.9 kg), followed by Case 2 (1031.7 kg), Case 5 (1081.8 kg), Case 6 (1095.8 kg), Case 3 (1118.2 kg), and Case 4 (1160.7 kg). With the same configuration of web members, the trapeizoidal truss is always lighter than the parallel chord truss. In three cases of trapeizoidal shape, the weight of the Warren truss (Case 1) is the smallest, followed by the Howe truss (Case 5), and the Pratt truss (Case 3). For three parallel chord trusses, the order is the same when the Warren truss has the smallest weight, followed by the Howe amd the Pratt trusses, respectively. Overall, among six common types, the trapeizoidal Warren configuration is the most suitable solution for pinned-pinned roof trusses.

5. Conclusions

This paper uses an effective algorithm called AdaBoost-DE to optimize steel roof trusses. The AdaBoost-DE adopts the machine learning classification technique to enhance the performance of the DE algorithm. In more detail, the DE is used as a search engine while the AdaBoost classifier serves as a coarse filter to remove poor trial vectors during the optimization process. The applicability of the AdaBoost-DE algorithm in the practical design is demonstrated through an example to design a steel roof truss of an industrial factory. The numerical example shows that the AdaBoost-DE achieves the same optimal design as the original DE algorithm but it reduces about 36% of fitness evaluations.

Obviously, the proposed AdaBoost-DE method requires additional times to train models. Therefore, this method is effective when solving problems having computationally expensive fitness evaluation, for example, large-scale structures where the training time is very small in comparison with the time of finite element analysis. In the future, the application of the AdaBoost-DE for optimizing other structures like double-layer grids, lattice towers can be investigated. Moreover, the study on integrating other classification techniques into evolutionary algorithms has great potential.

Acknowledgments

The first author was funded by Vingroup Joint Stock Company and supported by the Domestic PhD. Scholarship Programme of Vingroup Innovation Foundation (VINIF), Vingroup Big Data Institute (VINBIGDATA) code VINIF.2020.TS.134.

References

[1] Schmit, L. A., Farshi, B. (1974). Some Approximation Concepts for Structural Synthesis. *AIAA Journal*, 12(5):692–699.

[2] Goldberg, D. E., Samtani, M. P. (1986). Engineering optimization via genetic algorithm. In *Electronic computation*, ASCE, 471–482.

[3] Cai, J., Thierauf, G. (1996). Evolution strategies for solving discrete optimization problems. *Advances in Engineering Software*, 25(2-3):177–183.

[4] Wang, Z., Tang, H., Li, P. (2009). Optimum Design of Truss Structures Based on Differential Evolution Strategy. In *2009 International Conference on Information Engineering and Computer Science*, IEEE.

[5] Perez, R. E., Behdinan, K. (2007). Particle swarm approach for structural design optimization. *Computers & Structures*, 85(19-20):1579–1588.

[6] Camp, C. V., Bichon, B. J. (2004). Design of Space Trusses Using Ant Colony Optimization. *Journal of Structural Engineering*, 130(5):741–751.

[7] Sonmez, M. (2011). Artificial Bee Colony algorithm for optimization of truss structures. *Applied Soft Computing*, 11(2):2406–2418.

[8] Degertekin, S. O. (2012). Improved harmony search algorithms for sizing optimization of truss structures. *Computers & Structures*, 92-93:229–241.

[9] Koumousis, V. K., Georgiou, P. G. (1994). Genetic Algorithms in Discrete Optimization of Steel Truss Roofs. *Journal of Computing in Civil Engineering*, 8(3):309–325.

[10] Croce, E. S., Ferreira, E. G., Lemonge, A. C., Fonseca, L. G., Barbosa, H. J. (2004). A genetic algorithm for structural optimization of steel truss roofs. In *In XXV CILAMCE, 25th Iberian Latin-American Congress on Computational Methods in Engineering, UFPE*, Recife, PE, Brasil.

[11] Hamza, K., Mahmoud, H., Saitou, K. (2003). Design optimization of N-shaped roof trusses using reactive taboo search. *Applied Soft Computing*, 3(3):221–235.

[12] Hieu, N. T., Tuan, V. A. (2018). Weight optimization of composite cellular beam based on the differential evolution algorithm. *Journal of Science and Technology in Civil Engineering (STCE) - NUCE*, 12(5): 28–38.

[13] Papadrakakis, M., Lagaros, N. D., Tsompanakis, Y. (1999). Optimization of Large-Scale 3-D Trusses Using Evolution Strategies and Neural Networks. *International Journal of Space Structures*, 14(3):211–223.

[14] Salajegheh, E., Gholizadeh, S. (2005). Optimum design of structures by an improved genetic algorithm using neural networks. *Advances in Engineering Software*, 36(11-12):757–767.

[15] Kaveh, A., Gholipour, Y., Rahami, H. (2008). Optimal Design of Transmission Towers Using Genetic Algorithm and Neural Networks. *International Journal of Space Structures*, 23(1):1–19.

[16] Nguyen, T.-H., Vu, A.-T. (2020). Using Neural Networks as Surrogate Models in Differential Evolution Optimization of Truss Structures. In *Computational Collective Intelligence*, Springer International Publishing, 152–163.

[17] Nguyen, T. H., Vu, A. T. (2021). Speeding up Composite Differential Evolution for structural optimization using neural networks. *Journal of Information and Telecommunication*, 1–20.

[18] Hung, T. V., Viet, V. Q., Thuat, D. V. (2019). A deep learning-based procedure for estimation of ultimate load carrying of steel trusses using advanced analysis. *Journal of Science and Technology in Civil Engineering (STCE) - HUCE*, 13(3):113–123.

[19] Truong, V. H., Pham, H. A. (2021). Support Vector Machine for Regression of Ultimate Strength of Trusses: A Comparative Study. *Engineering Journal*, 25(7):157–166.

[20] Truong, V.-H., Vu, Q.-V., Thai, H.-T., Ha, M.-H. (2020). A robust method for safety evaluation of steel trusses using Gradient Tree Boosting algorithm. *Advances in Engineering Software*, 147:102825.

[21] Kim, S.-E., Vu, Q.-V., Papazafeiropoulos, G., Kong, Z., Truong, V.-H. (2020). Comparison of machine learning algorithms for regression and classification of ultimate load-carrying capacity of steel frames. *Steel and Composite Structures*, 37(2):193–209.

[22] Nguyen, T.-H., Vu, A.-T. (2022). Application of Artificial Intelligence for Structural Optimization. In *Modern Mechanics and Applications*, Springer, 1052–1064.

[23] Nguyen, T.-H., Vu, A.-T. (2021). A Comparative Study of Machine Learning Algorithms in Predicting the Behavior of Truss Structures. In *Research in Intelligent and Computing in Engineering*, Springer Singapore, 279–289.

[24] Nguyen, T.-H., Vu, A.-T. (2021). Evaluating structural safety of trusses using Machine Learning. In *Frattura ed Integrità Strutturale*, 308–318.

[25] TCVN 2737:1995. *Loads and Actions - Design Standard.* Ministry of Science and Technology.

[26] TCVN 5575:2012. *Steel Structures - Design Standard.* Ministry of Science and Technology.

[27] Storn, R., Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

[28] Freund, Y., Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139.